## REMARKS

The foregoing amendments have been requested to be entered to correct various errors in the application as originally filed. In addition, a brief description of FIGURES 7 and 8 were inadvertently left out in the originally-filed version of the application, and have been added herein.

*Charge Deposit Account*

Please charge our Deposit Account No. 02-2666 for any additional fee due in this matter.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR and ZAFMAN

Dated: Nov 1, 2001

R. Alan Burnett
Reg. No. 46,149

**FIRST CLASS CERTIFICATE OF MAILING**

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

on November 1, 2001
Date of Deposit

Jenny E. Miller
Name of Person Mailing Correspondence

Signature

11-01-01
Date

VERSION OF AMENDMENTS WITH MARKINGS TO SHOW CHANGES MADE

In the Specification:

The paragraph beginning at page 4, line 8 has been amended as follows:

FIGURE 6 is a flowchart illustrating the logic used by the invention when registering and installing event handlers that are stored in firmware volumes that are scanned during a pre-boot process; [and]

The following paragraphs have been added immediately following the preceding paragraph:

-- FIGURE 7 is a flowchart illustrating operations performed by the invention when registering event handlers for servicing processor management interrupt (PMI) event with an Itanium™ processor;

FIGURE 8 is a flowchart illustrating operations performed by the invention when handler a PMI event; and --

The paragraph beginning at page 4, line 21 has been amended as follows:

FIGURE [7] 9 is a schematic diagram of a person computer system suitable for implementing the present invention.

The paragraph beginning at page 14, line 21 has been amended as follows:

Next, in a decision block 108 a determination is made to whether enough SMRAM is available to hold the event handler routine. If not enough SMRAM memory space is available, the logic proceeds to a block 110 in which the caller is alerted. As an option, in response to being alerted, the caller may use the SMM_ACCESS::GetCapabilities and SMM_ACCESS::AcquireSmramRange method to acquire additional memory space within the SMRAM, as provided by a block 113. If there is not enough SMRAM memory space available, the SMRAM is closed by calling the SMM_ACCESS::Close method and an error code is returned to the caller in an error return block 114.

The paragraph beginning at page 15, line 26 has been amended as follows:

With reference to FIGURE 6, the mechanism begins in a block 130, wherein the SMM_BASE driver searches all firmware volumes that are materialized in the system during pre-boot. As defined by start and end loop blocks 132 and 134, the following logic is applied to each of these firmware volumes. In a decision block 136 a determination is made to whether the firmware volume contains any firmware files conformant with the firmware file system. If the answer is no, the logic loops back to examine the next firmware volume. If one or more conformant firmware files are found, each of these files are examined using the following process, as defined by start and end loop blocks 138 and 140. In a decision block 142, the SMM_BASE drive examines the file type of the current file to determine with it is an "SMMHandler" file. If it is not, the logic loops back to begin examination of the next file. If the file type is "SmmHandler," the SMM_BASE driver decomposes the *Sections* of the firmware file in a block 144; a section is the internal packing mechanism within a firmware file. As provided by a block [136] 146, if a section contains a PE32+ executable image, wherein PE32+ is a Portable Executable image type described by Microsoft in the Portable Image specification (posted on the Internet at "www.microsoft.com/hwdev/efi") that is of the same machine type as which the SMM_BASE is implemented (e.g., the computer system is an IA32 machine and the handler is an IA32 PE32+ image) or if the SMM_BASE implementation is on an IA32 system that supports loading legacy 16-bit handlers, the SMM_BASE driver shall install the executable image or legacy 16-bit handler contained in the section [in a block 146]. The logic then proceeds to process subsequent firmware files and firmware volumes in a similar manner.

In the Claims:

Claim 8 has been amended as follows:

8. (Amended) [A] The method [for] of claim 7, wherein each of said plurality of event handlers comprise a set a machine code that is executed by the processor

to service an error condition generated by a hardware component in the computer system that causes the event, and determining whether an event handler is the appropriate event handler for servicing the event comprises:

executing a first portion of the set of machine code corresponding to the event handler that was most recently dispatched that queries the hardware component corresponding to that event handler to determine if the error condition was caused by that hardware component; and

completing execution of the set of machine code for the event handler if it is determined that the error condition was cause by its corresponding hardware component, otherwise returning a value to the event handler management service indicating that the event handler is not the appropriate event handler to service the error condition.

Claim 16 has been amended as follows:

16. Amended) The method of claim 14, further comprising authenticating the event handler before it is loaded into [SRMAM] SMRAM.